NONPROCEDURAL IMPLEMENTATION OF MATHEMATICAL PROGRAMMING ALGORITHMS [*]

JAMES K. HO
Management Science Program, University of Tennessee
Knoxville, Tennessee 37996

ABSTRACT

It is shown that the implementation of many mathematical programming algorithms using spreadsheet software can be described as instances of a nonprocedural approach to computer programming. This concept is generalized to an abstract computing environment called HYPERCELLS. Its significance in view of rapid development in parallel computation is discussed.

1. INTRODUCTION

With the widespread use of microcomputers in academia, business, industry and government, spreadsheet software has become the most popular general purpose quantitative tool for numerical computation. It is essentially a two dimensional array of cells. Each cell may contain text for documentation, numerical values from data, or a formula dependent on values in other cells. Whenever the value in a cell is changed, the content of cells with formulas depending on it can be recalculated automatically. A simple example is given in Figure 1. Cells B1 through B4 contain quarterly revenues. Cell B6 contains a formula for the sum of the values in B1 through B4. It therefore represents the total revenue for the year. When the quarterly values are entered or altered, the total is updated accordingly.

Most commercial spreadsheet packages provide a substantial set of mathematical functions with which complex formulas can be written for nontrivial computational models. Specifics of a design include the order of computation and the ability to resolve circular references. For this reason, while the initial thrust has been in finance and accounting applications, there is a growing interest in this type of computing environment in science and engineering.

In this paper, we demonstrate and generalize certain abstract properties of modeling and computing in the spreadsheet environment. They can be interpreted as a nonprocedural approach to the implementation of numerical methods in general and mathematical programming in particular. A nonprocedural approach to computer programming is one that relies more on the characterization of the solution than on the description of every step

* To appear in Impact of Recent Computer Advances on Operations Research, Elsevier-North Holland (1989)

required in the computation. The distinction from conventional computer programming is shown in Section 2. The cases of dynamic and linear programming are presented in Sections 3 and 4 respectively. Section 5 introduces the concept of Hypercells as a general abstraction of the spreadsheet and discusses the potential of parallel processing within this framework.

The purpose of this paper is to formalize to some extent the departure from conventional implementation of mathematical programming afforded by the spreadsheeet environment. Its significance draws from both the ever increasing popularity of the latter as well as the potential for generalization. At present it is difficult to project future development in this direction and hence no attempt will be made to compare the absolute effectiveness of the different approaches.

	A	B
1	First Quarter	$125,000.00
2	Second Quarter	$145,000.00
3	Third Quarter	$137,000.00
4	Fourth Quarter	$158,000.00
5		
6	Total Revenue	$565,000.00

Figure 1. Simple Example of a Spreadsheet Model

	A	B	C	D	E	F	G	H	I	J	K	L
1	1	3	-2	-1		3	3	0		3	-4	1
2	0	-1	0	3	x	0	-1	2	=	6	1	7
3	2	4	1	2		-1	2	1		9	4	15
4						2	0	3				

Figure 2. Nonprocedural Implementation of Matrix Multiplication

2. NONPROCEDURAL IMPLEMENTATION OF NUMERICAL METHODS

To illustrate the nonprocedural environment provided by spreadsheet software, we use the simple example of matrix multiplication. In Figure 2, the range of cells from A1 to D3 contains a 3 by 4 matrix A. The range of cells from F1 to H4 contains a 4 by 3 matrix B. The product C is in the range from J1 to L3.

In conventional computer programming, a procedure can be coded to compute the coefficients of the product matrix. Indeed, by looping over i from 1 to 3 and j from 1 to 3, C_{ij} is given by the inner product of row i in A and Column j in B. The formula for the inner product is specified once with generic indices i and j and executed repeatedly as i and j take on different values over the appropriate ranges.

Consider next the situation on a spreadsheet. The value of C_{11} is given by a formula in cell J1: +\$A1*F\$1+\$B1*F\$2+\$C1*F\$3+\$D1*F\$4. The "\$" sign indicates an absolute address and its absence indicates a relative address. Since all commercial spreadsheet packages include such feature, this formula can be copied to all the other cells in C using typically a single operation. Each cell in C will then contain a formula for the appropriate inner product. When entries in A or B are altered, the values in C will be recalculated. Note that ultimately, the values in C are computed "procedurally" in some well defined sequence specific to the particular spreadsheet environment. What we mean here by a nonprocedural approach is that there is no explicit code in some computer language for the implementation of the algorithm for matrix multiplication. Instead, the spreadsheet in Figure 2 contains the essence of the operation, namely, that every coefficient in the result is the inner product of a row and a column. It is quite obvious that this nonprocedural approach implies redundancy and requires more memory than conventional programming. However, this redundancy can be exploited in parallel computation with multiprocessers. In our example, each inner product for the coefficient of the product matrix can be computed independently.

Using the same principle, many numerical methods can be implemented on a spreadsheet. Examples of significant potential include finite difference methods for the solution of partial differential equations. In the following, we focus on mathematical programming and illustrate the nonprocedural approach for both dynamic and linear programming.

3. DYNAMIC PROGRAMMING

Consider the simple example in Figure 3. There are two stages. Stage 2 has two states: C and D. State C has a single choice with resulting state E and immediate payoff of 3 units. State D has a single choice with resulting state E and immediate payoff of 2 units. Stage one has two states: A and B. State A has two choices: one with resulting state C and immediate payoff of 2 units; the other with resulting state D and immediate payoff of 5 units. State B has two choices: one with resulting state C and immediate payoff of 3 units; the other with resulting state D and immediate payoff of 4 units. The DP problem is to find an optimal policy which prescribes a sequence of choices from each starting state in Stage 1 to termination that will maximize the total payoff.

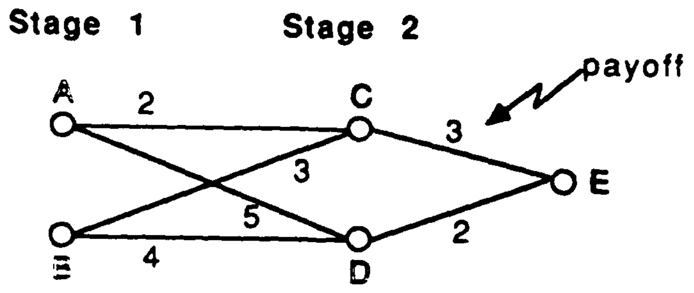

Figure 3. A Simple Example in Dynamic Programming

	A	B	C	D	E	F	G
1	# of stages		STAGE	1	# of states	2	
2	2		State	# of decisions	Opt. Policy	Opt. Value	
3			A	2	D	7	
4			B	2	alternatives...	6	
5							
6			Stage	1	State	A	
7			Decision	Payoff now	Payoff later	Total Payoff	
8			C	2	3	5	
9			D	5	2	7	*
10							
11			Stage	1	State	B	
12			Decision	Payoff now	Payoff later	Total Payoff	
13			C	3	3	6	*
14			D	4	2	6	*

	H	I	J	K	L
1	STAGE	2	# of states	2	
2	State	# of decisions	Opt. Policy	Opt. Value	
3	C	1	E	3	
4	D	1	E	2	
5					
6	Stage	2	State	C	
7	Decision	Payoff now	Payoff later	Total Payoff	
8	E	3	0	3	*
9					
10	Stage	2	State	D	
11	Decision	Payoff now	Payoff later	Total Payoff	
12	E	2	0	2	*
13					
14					

Figure 4. Spreadsheet Implementation of a Dynamic Programming Model

The method of recursion is assumed well known and will not be described here. To implement this method on a spreadsheet, it is necessary to design a format for the DP model. Figure 4 represents the above example in the format used in [Ho 1987]. There is a section with five columns for each stage. The top part of such a section contains the optimal policy for the stage. For instance, from State A at Stage 1, the optimal choice is to go to State D in the next stage with an optimal value of 7. Whereas from State B there are alternative optimal choices with a value of 6. The rest of the section contains the choices for each state. A decision is specified by the resulting state at the next stage. The payoff now is problem data. The payoff later is given by the optimal value of the resulting state. The total payoff is the sum of payoff now and payoff later. The optimal value for the state is the maximum of the total payoffs among the choices. An asterisk indicates an optimal choice. The optimal policy is the optimal choice if unique, otherwise there are alternatives indicated by multiple asterisks

The above relations in the DP model are constructed recursively from the last stage backwards. The actual formulas may depend on the particular spreadsheet environment. In any case, it is possible to implement the DP method nonprocedurally so that when immediate payoffs are changed, the solution will be recomputed automatically. Dynamic programming recursion is perhaps the simplest and most natural example of the nonprocedural implementation of optimization algorithms on spreadsheets.

4. LINEAR PROGRAMMING

Both the algebraic formulation of linear programs and the simplex method [Dantzig 1963] have natural tabular formats. On a spreadsheet, one can encode the definition of the simplex tableau. Using a considerably more intricate and elaborate design of a recursive system than that for DP, the tableau can be updated automatically until the stopping criterion is met. In the following, we will describe a particular implementation in some detail.

Consider the following linear programming problem.

$$\begin{array}{lrcrcrcl}
\text{Maximize} & 8A & + & 11B & + & 9.5\,C & & \\
\text{Subject to} & 3A & + & 2B & + & 4C & \leq & 74 \\
 & A & + & 2B & & & \leq & 40 \\
 & 2A & & & + & 3C & \leq & 50 \\
 & & & B & & & \leq & 10 \\
 & & & A,B,C & \geq & 0 & &
\end{array}$$

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	0	Step:	0	Status:	Phase 2								
2										If Piv	PIVC		
3	PIVC:		0	0	0	0	0	1	0	1	6		
4	j:		1	2	3	4	5	6	7	8	Basis		
5													
6	Pivot Row		0	0	0	0	0	0	0	10			
7													
8	i:	PivCol								RHS		Ratios	Dj
9	-1	0	0	0	0	0	0	0	0	0			0
10	0	11	0	0	0	0	8	11	9.5	0			11
11	1	2	1	0	0	0	3	2	4	74	1	37	0
12	2	2	0	1	0	0	1	2	0	40	2	20	0
13	3	0	0	0	1	0	2	0	3	50	3	•••••	0
14	4	1	0	0	0	1	0	1	0	10	4	10	4
15													
16												MinRat	PIVR
17												10	4
18												PIVOT	1

Figure 5. Initial Tableau for LP Example

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	1	Step:	3	Status:	Optimal								
2										If Piv	PIVC		
3	PIVC:		0	0	0	0	0	0	0	0	None		
4	j:		1	2	3	4	5	6	7	8	Basis		
5													
6	Pivot Row		0.25	0	0	-0.5	0.75	0	1	13.5			
7													
8	i:	PivCol								RHS		Ratios	Dj
9	-1	0	0	0	0	0	0	0	0	0			0
10	0	-1.16	-2.66	0	0	-5.66	0	0	-1.16	-254			0
11	1	1.333	0.333	0	0	-0.66	1	0	1.333	18	5	13.5	1
12	2	-1.33	-0.33	1	0	-1.33	0	0	-1.33	2	2	•••••	0
13	3	0.333	-0.66	0	1	1.333	0	0	0.333	14	3	42	0
14	4	0	0	0	0	1	0	1	0	10	6	•••••	0
15													
16												MinRat	PIVR
17												13.5	1
18												PIVOT	1.333

Figure 6. Final Tableau for LP Example

Figure 5 illustrates a spreadsheet implementation of the simplex tableau. Initially, this represents the original problem in standard form. However, the formulas involved are defined in such a way that subsequent recalculations lead eventually to the final tableau shown in Figure 6. The required formulas are tabulated below in generic, pseudo-code format so that we are not confined to the specifics of particular software.

Cell Range	Content	Formula
A1	Mode	0 to initialize; 1 thereafter
C1	Iteration Count	If Mode=0 then 0, else Iteration Count+1
F1	Solution Status	If SumOfInfeasibilities>0 then
		if pivot then "Phase 1"
		else "Infeasible"
		else if MaxReducedCost≤0 then "Optimal"
		else if pivot then "Phase 2"
		else "Unbounded"
C3..I3	Pivot Column Selector	If SumOfInfeasibilities>0 then
		if Phase1ReducedCost of Column =
		MaxPhase1ReducedCost and
		MaxPhase1ReducedCost > 0 then
		Value=1
		else Value=0
		else if Phase2ReducedCost of Column =
		MaxPhase2ReducedCost and
		MaxPhase2ReducedCost > 0 then
		Value=1
		else Value=0
J3	Pivot Column Indicator	Max(C3..I3)
K3	Pivot Column Index	If PivotColumnIndicator=0 then
		Value="None"
		else Value=index of first column with Pivot
		Column Selector of 1
C4..J4	Column Indices	j=1,...,8
C6..J6	Pivot Row	A vertical look-up of the pivot row
A9..A14	Row Indices	i= -1 for Phase 1 Objective
		i= 0 for Phase 2 Objective
		i = 1,...,4 for constraints
B9..B14	Pivot Column	A horizontal look-up of the pivot column
C9..F9	Phase 1 Reduced Costs	If Mode>0 then NewValue=Simplex Pivot
	for Logical Variables	Update of OldValue (using the pivot
		and appropriate entries in the pivot row
		and pivot column)
		else if Logical Variable is in an " =" row

		then Value=0 else Value=Partial Sum of column corresponding to pivoting on artificials
G9..I9	Phase 1 Reduced Costs for Structural Variables	If Mode>0 then NewValue=Simplex Pivot Update of OldValue else Value=Partial Sum of column corresponding to pivoting on artificials
J9	Sum of Infeasibilities	Same as above
C10..F10	Phase 2 Reduced Costs for Logical Variables	If Mode>0 then if Logical Variable is in an " =" row then Value=0 else NewValue=Simplex Pivot Update of OldValue else Value=0
G10..I10	Phase 2 Reduced Costs for Structural Variables	If Mode>0 the NewValue=Simplex Pivot Update of OldValue else if Minimization then Value= - Objective Coefficient of Structural Variable else Value=Objective Coefficient of Structural Variable
J10	Phase 2 Objective Value	If Mode>0 then NewValue=Simplex Pivot Update of OldValue else Value = 0

C11,D12,....,F14	Diagonal entries in Tableau for Logical Variables	If Mode>0 then
		if Row is Pivot Row then NewValue =
		OldValue/Pivot
		else NewValue=Simplex Pivot Update
		of OldValue
		else if RightHandSide≥0 then
		if not "≥" constraint then Value=1
		else Value= -1
		else if "≤" constraint then Value= -1
		else Value=1

C11..F14 except above		

	Off-diagonal entries in Tableau for Logical Variables	If Mode>0 then
		    if Row is Pivot Row then NewValue =
		        OldValue/Pivot
		    else NewValue=Simplex Pivot Update
		        of OldValue
		else Value=0

G11..I14	Tableau for Structural Variables	If Mode>0 then
		    if Row is Pivot Row then NewValue =
		        OldValue/Pivot
		    else NewValue=Simplex Pivot Update
		        of OldValue
		else if RightHandSide>0 then
		        Value=Matrix Coefficient
		    else Value= - Matrix Coefficient

J11..J14	Right-Hand-Side of Tableau	If Mode>0 then
		    if Row is Pivot Row then NewValue =
		        OldValue/Pivot
		    else NewValue=Simplex Pivot Update
		        of OldValue
		else Value=Absolute Value of
		    RightHandSide

M9	Maximum Phase 1 Reduced Cost	If Max(C9..I9)>0 then
		    Value= Max(C9..I9)
		else Value=0

M10	Maximum Phase 2 Reduced Cost	If Max(C10..I10)>0 then
		    Value= Max(C10..I10)
		else Value=0

K11..K14	Basis Indices (Index of Column basic in Row)	If Mode>0 then
		    if row is Pivot Row then
		        NewValue=Index of Pivot Column
		    else NewValue=OldValue
		else if RightHandSide≥0 and "≤" constraint
		        then Value=Row Index
		    else if RightHandSide<0 and "≥" const-
		            raint then Value=Row Index
		        else Value= -Row Index

L11..L14	Pivot Ratios	If nonbinding row then Value=Infinity else if TableauEntry>Pivot Tolerance and RightHandSide≥0 then Value= RightHandSide /TableauEntry else if TableauEntry< Zero Tolerance and basic column is artificial and RightHandSide< Zero Tolerance then Value=0 else Value=Infinity
M11..M14	Min Ratio Indicator	If Ratio=MinRatio and MinRatio<Infinity then Value=Row Index else Value=0
L17	Min Ratio	MIN(L11..L14)
M17	Pivot Row Index	MAX(M11..M14)
M18	Pivot	If PivotRow=0 then Value="none" else Value=Pivot Coefficient in Tableau

Note that the above spreadsheet template is a complete implementation of the two-phase simplex method for general LP problems with any type of constraints and right-hand-sides. A similar approach that is less compact and less general first appeared in [Carroll 1986]. To gain this compactness in the present design, it is necessary to recalculate various ranges of the spreadsheet separately. This can be automated by a few lines of macro instructions of the following form.

```
While PivotColumnIndex≠"None" and PivotRowIndex≠"None"
    Recalculate (A1..M18)          {Update Tableau}
    Recalculate (C3..K3)           {Find pivot column}
    Recalculate (B9..B14)
    Recalculate (L11..L17)         {Find pivot row}
    Recalculate (M9..M18)
```

The basic difference between the spreadsheet implementation and, say, a conventional Fortran code of the simplex method, or one coded in a spreadsheet macro language [Ho 1987] is that the former emphasizes the actual definition of the important concepts whereas the latter deals with their translation into computational procedures.

Essentially the spreadsheet in Figure 5 says that the simplex tableau is the update of itself. The definition of the update is incorporated directly into the formulas for the cells holding th. tableau, just like the defintition of matrix multiplication is expressed in our earlier example. The process of constructing the entire LP spreadsheet can also be automated with e.g. macro programs that define all the appropriate ranges and formulas according to given dimensions. Functional versions of such software using Lotus 1-2-3 [Lotus Development Corporation 1985] are used in this work. While we are not concerned with direct comparisons of computational efficiency at this stage, it is clear that memory usage in the nonprocedural approach is going to be high because of the replication of formulas. However, such redundancy may be exploited eventually in parallel computations as we shall discuss in some generality in the next section. In terms of solution speed, it is significantly more efficient than a macro program for the simplex method [Ho 1987]. To date, our initial experience has been with small textbook problems with less than 50 constraints.

5. HYPERCELLS AND PARALLEL COMPUTATION

The spreadsheet belongs to a new paradigm in computing environments that is still in its infancy. Our effort is to illustrate its implications in mathematical programming rather than to extol its actual efficacy. Critics who hasten to challenge the significance of this association should be reminded of the case with parallel computing. There too, one often found it difficult to justify radical approaches for incremental improvements. However, as multiprocessor computers become prevalent, a constructive rethinking of computational methodology begins to take shape. Along such lines, we conclude the paper with a vision of the nonprocedural environment.

Consider the n-dimensional generalization of the two-dimensional spreadsheet. It is an array of n-tuples that we shall call hypercells. Each hypercell may contain text, a value or a function in values in other hypercells. Functions are defined in a given library and should include all the usual mathematical and logical operations. A protocol for the recalculation of functional values is also given.

As an example of the possible application of such a computing environment, we can extend the above optimization models and have a third dimension representing changes in parameters as a function of discrete time points. The nonprocedural implementation of linear or dynamic programming can then provide time-phased solutions to a sequence of related problems. At this writing, the forthcoming version of a popular spreadsheet software package is already slated to be three-dimensional.

With the rapid development of parallel computer architecture (see e.g. [Fox and

Messina 1987]), it will also become attractive to distribute the tasks of recalculating the hypercells over an array of processors. The fact that formulas are explicitly associated with each hypercell in the nonprocedural implementation of numerical methods makes it especially suitable for parallel processing. The main concern with programming in this environment will be the control of the interdependencies of data with respect to the given protocol for recalculation. In cases where most of the hypercells can be updated independently and concurrently, like most entries in our LP tableau, this parallel nonprocedural approach should be very efficient. While the performance of single processors are bounded by physical limits, their costs are continuing to lower. For this reason, a multiprocessing hypercell computing environment should become viable in the foreseeable future.

Acknowledgement

The author wishes to thank T. Owen Carroll for a stimulating discussion. This research was supported in part by the Office of Naval Research under grant N00014-87-K-0163.

References

Carroll, T.O. (1986), *Decision Power with Supersheets*, Dow Jones-Irwin, Homewood, IL.

Dantzig, G.B.(1963), *Linear Programming and Extensions*, Princeton University Press, Princeton, NJ.

Fox, G.C. and P.C. Messina (1987) "Advanced Computer Architectures", *Scientific American* Vol 257, No 4.

Ho, J.K. (1987), *Linear and Dynamic Programming with Lotus 1-2-3* , MIS Press, Portland, OR.

Ho, J.K., T.C. Lee and R.P. Sundarraj (1988), "Decomposition of Linear Programs using Parallel Computation", *Mathematical Programming* (in press.)

Lotus Development Corporation (1985), *Lotus 1-2-3 Reference Manual*, Cambridge, MA.

ADA203392

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER MSP-88-1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) Nonprocedural Implementation of Mathematical Programming Algorithms		5. TYPE OF REPORT & PERIOD COVERED Technical Report F/88
		6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) James K. Ho		8. CONTRACT OR GRANT NUMBER(s) N00014-87-K-0163
9. PERFORMING ORGANIZATION NAME AND ADDRESS College of Business Admin., University of Tenn. Department of Management, 615 Stokely Mgmt Ctr Knoxville, TN 37996		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Department of the Navy 800 N. Quincy Street Arlington, VA 22217-5000		12. REPORT DATE December 1988
		13. NUMBER OF PAGES 12 pages
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

This documentation has been approved for public release and sale, its distribution is unlimited.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Mathematical Programming
Spreadsheet Software
Parallel Computing
Nonprocedural Programming

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

It is shown that the implementation of many mathematical programming algorithms using spreadhseet software can be described as instances of a nonprocedural approach to computer programming. This concept is generalized to an abstract computing environment called HYPERCELLS. Its significance in view of rapid development in parallel computation is discussed.